First Hit Fwd Refs

Generate Collection Print

L1: Entry 211 of 314 File: USPT May 16, 2000

US-PAT-NO: 6064812
DOCUMENT-IDENTIFIER: US 6064812 A

TITLE: System and method for developing automation clients using a graphical data flow program

DATE-ISSUED: May 16, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Parthasarathy; Murali	Austin	TX		
Sojoodi; Omid	Austin	TX		

US-CL-CURRENT: 717/105; 717/116, 717/163

CLAIMS:

We claim:

1. A computer-implemented method for creating a graphical program which is an automation client, wherein the graphical program is operable to invoke a method of an object, wherein the method for creating the graphical program operates in a computer including a display screen and a user input device, wherein the object is exported by an automation server, the method for creating the graphical program comprising:

receiving user input specifying an automation class;

displaying on the screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

displaying on the screen an automation invoke node in response to user input, wherein said automation invoke node is operable to invoke a method of said object;

connecting said automation open node to said automation invoke node in response to user input, wherein said automation open node is operable to provide information on said object to said automation invoke node;

wherein said automation invoke node receives said information on said object, wherein said automation invoke node is operable to invoke said method of said object during execution of said graphical program.

2. The computer-implemented method of claim 1, further comprising:

constructing execution instructions in response to said graphical automation client, wherein said execution instructions are executable to instantiate an object of said automation class and invoke said method of said object.

3. The computer-implemented method of claim 2, further comprising:

executing said execution instructions, wherein said automation invoke node invokes said method of said object during said executing.

4. The computer-implemented method of claim 1, further comprising:

displaying on the screen a list of type libraries associated with one or more automation servers present in the computer;

selecting a type library from said list of type libraries in response to said receiving user input specifying said automation class, wherein said automation class is from said selected type library.

5. The computer-implemented method of claim 1, further comprising:

displaying on the screen a list of possible automation classes from a type library associated with said automation server;

selecting said automation class from said list of possible automation classes in response to said receiving user input specifying said automation class.

6. The computer-implemented method of claim 1, wherein said information on said object includes a reference to said object.

7. The computer-implemented method of claim 1, wherein said displaying on the screen said automation invoke node comprises selecting said method in response to user input and displaying on the screen said method in said automation invoke node.

8. The computer-implemented method of claim 1, wherein said method includes one or more parameters, wherein said displaying on the screen said automation invoke node comprises displaying a terminal for each of said one or more parameters.

9. The computer-implemented method of claim 8, wherein each said terminal for each of said one or more parameters has an associated data type.

10. The computer-implemented method of claim 1, wherein said automation class is an automation class of an automation technology selected from the list of automation technologies consisting of OLE, CORBA and OpenDoc.

11. The computer-implemented method of claim 1, further comprising:

displaying a list of automation technologies;

selecting one of said automation technologies, in response to said receiving user input specifying

said automation class, wherein said automation class conforms with said selected automation technology.

12. The computer-implemented method of claim 1, further comprising:

displaying a palette of automation nodes from which said automation open node and said automation invoke node may be selected.

13. The computer-implemented method of claim 1, wherein the graphical program is operable to invoke said method of said object for performing instrumentation functions on an instrument.

14. The computer-implemented method of claim 1, wherein said connecting said automation open node to said automation invoke node comprises displaying on the screen a wire connecting said automation open node to said automation invoke node.

15. A computer-implemented method for creating a graphical program which is a client, wherein the graphical program is operable to invoke a method of an object, wherein the method for creating the graphical program operates in a computer including a display screen and a user input device, wherein the object is exported by a server, the method for creating the graphical program comprising:

receiving user input specifying a class;

displaying on the screen an open node in response to user input, wherein said open node is operable to create an instance of said class, wherein said instance of said class is an object;

displaying on the screen an invoke node in response to user input, wherein said invoke node is operable to invoke a method of said object;

connecting said open node to said invoke node in response to user input, wherein said automation open node is operable to provide information on said object to said invoke node;

wherein said invoke node receives said information on said object, wherein said invoke node is operable to invoke said method of said object during execution of said graphical program.

16. A computer-implemented method for creating a graphical automation client in a graphical program which is operable to invoke a method of an object, wherein the method for creating the graphical program operates in a computer including a display screen and a user input device, wherein the object is exported by an automation server, the method for creating the graphical program comprising:

displaying on the screen a reference to an automation class in response to user input;

displaying on the screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

connecting said automation open node to said automation class reference in response to user input, wherein said automation class reference is operable to provide information on said automation class to said automation open node, wherein said automation open node uses said information on

said automation class received from said automation class reference to instantiate said object;

displaying on the screen an automation invoke node in response to user input, wherein said automation invoke node is operable to invoke a method of said object;

connecting said automation open node to said automation invoke node in response to user input, wherein said automation open node is operable to provide information on said object to said automation invoke node;

wherein said automation invoke node receives said information on said object, wherein said automation invoke node is operable to invoke said method of said object during execution of said graphical program.

17. The computer-implemented method of claim 16, wherein said connecting said automation open node to said automation class reference comprises displaying on the screen a wire connecting said automation open node to said automation class reference.

18. A computer-implemented method for creating a graphical program which is an automation client, wherein the graphical program is operable to invoke a property of an object, wherein the method for creating the graphical program operates in a computer including a display screen and a user input device, wherein the object is exported by an automation server, the method for creating the graphical program comprising:

receiving user input specifying an automation class;

displaying on the screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

displaying on the screen an automation property node in response to user input, wherein said automation property node is operable to invoke a property of said object;

connecting said automation open node to said automation property node in response to user input, wherein said automation open node is operable to provide information on said object to said automation property node;

wherein said automation property node receives said information on said object, wherein said automation property node is operable to invoke said property of said object during execution of said graphical program.

19. The computer-implemented method of claim 18, further comprising:

constructing execution instructions in response to said graphical automation client, wherein said execution instructions are executable to instantiate an object of said automation class and invoke said property of said object.

20. The computer-implemented method of claim 19, further comprising:

executing said execution instructions, wherein said automation invoke node invokes said property on said object during said executing.

21. The computer-implemented method of claim 18, further comprising:

displaying on the screen a list of type libraries associated with one or more automation servers present in the computer;

selecting a type library from said list of type libraries, in response to said receiving user input specifying said automation class, wherein said automation class is from said selected type library.

22. The computer-implemented method of claim 18, further comprising:

displaying on the screen a list of possible automation classes from a type library associated with said automation server;

selecting said automation class from said list of possible automation classes in response to said receiving user input specifying said automation class.

23. The computer-implemented method of claim 18, wherein said information on said object includes a reference to said object.

24. The computer-implemented method of claim 18, wherein said displaying on the screen said automation property node comprises selecting said property in response to user input and displaying on the screen said property in said automation property node.

25. The computer-implemented method of claim 18, wherein said displaying on the screen said automation property node comprises displaying a terminal for said property.

26. The computer-implemented method of claim 25, wherein said terminal has an associated data type.

27. The computer-implemented method of claim 18, wherein said automation class is an automation class of an automation technology selected from the list of automation technologies consisting of OLE, CORBA and OpenDoc.

28. The computer-implemented method of claim 18, further comprising:

displaying a list of automation technologies;

selecting one of said automation technologies, in response to said receiving user input specifying said automation class, wherein said automation class conforms with said selected automation technology.

29. The computer-implemented method of claim 18, further comprising:

displaying a palette of automation nodes from which said automation open node and said automation property node may be selected.

30. The computer-implemented method of claim 18, wherein the graphical program is operable to invoke said property of said object for performing instrumentation functions on an instrument.

31. The computer-implemented method of claim 18, wherein said connecting said automation open node to said automation property node comprises displaying on the screen a wire connecting said automation open node to said automation property node.

32. A computer-implemented method for creating a graphical program which is a client, wherein the graphical program is operable to invoke a property of an object, wherein the method for creating the graphical program operates in a computer including a display screen and a user input device, wherein the object is exported by an server, the method for creating the graphical program comprising:

receiving user input specifying a class;

displaying on the screen an open node in response to user input, wherein said open node is operable to create an instance of said class, wherein said instance of said class is an object;

displaying on the screen a property node in response to user input, wherein said property node is operable to invoke a property of said object;

connecting said open node to said property node in response to user input, wherein said open node is operable to provide information on said object to said property node;

wherein said property node receives said information on said object, wherein said property node is operable to invoke said property of said object during execution of said graphical program.

33. A computer-implemented method for creating a graphical automation client in a graphical program which is operable to invoke a property of an object, wherein the method operates in a computer including a display screen and a user input device, wherein the object is exported by an automation server, the method comprising:

displaying on the screen a reference to an automation class in response to user input;

displaying on the screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

connecting said automation open node to said automation class reference in response to user input, wherein said automation class reference is operable to provide information on said automation class to said automation open node, wherein said automation open node uses said information on said automation class received from said automation class reference to instantiate said object;

displaying on the screen an automation property node in response to user input, wherein said automation property node is operable to invoke a property of said object;

connecting said automation open node to said automation property node in response to user input, wherein said automation open node is operable to provide information on said object to said automation property node;

wherein said automation property node receives said information on said object, wherein said automation property node is operable to invoke said property of said object during execution of

said graphical program.

34. The computer-implemented method of claim 33, wherein said connecting said automation open node to said automation class reference comprises displaying on the screen a wire connecting said automation open node to said automation class reference.

35. A computer-implemented method for creating a graphical program which is an automation client, wherein the graphical program is operable to invoke a function of an object, wherein the method operates in a computer including a display screen and a user input device, wherein the object is exported by an automation server, the method comprising:

receiving user input specifying an automation class;

displaying on the screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

displaying on the screen an automation function node in response to user input, wherein said automation function node is operable to invoke a function of said object;

connecting said automation open node to said automation function node in response to user input, wherein said automation open node is operable to provide information on said object to said automation function node;

wherein said automation function node receives said information on said object, wherein said automation function node is operable to invoke said function of said object during execution of said graphical program.

36. The computer-implemented method of claim 35, wherein said automation function node comprises an automation invoke node and said function comprises a method of said object.

37. The computer-implemented method of claim 35, wherein said automation function node comprises an automation property node and said function comprises a property of said object.

38. The computer-implemented method of claim 35, further comprising:

displaying on the screen a list of possible automation classes from a type library associated with said automation server;

selecting said automation class from said list of possible automation classes in response to said receiving user input specifying said automation class.

39. The computer-implemented method of claim 35, wherein said displaying on the screen said automation function node comprises selecting said function in response to user input and displaying on the screen said function in said automation function node.

40. A system for creating a graphical automation client in a graphical program which is operable to invoke a method of an object exported by an automation server, comprising:

a computer including a display screen;

a user input device for receiving user input specifying an automation class;

an automation open node which is displayed on the screen in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

an automation invoke node displayed on the screen in response to user input, wherein said automation invoke node is operable to invoke a method of said object;

a wire which is displayed on the screen in response to user input for connecting said automation open node to said automation invoke node, wherein said automation open node is operable to provide information on said object to said automation invoke node;

wherein said automation invoke node receives said information on said object, wherein said automation invoke node is operable to invoke said method of said object during execution of said graphical program.

41. The system of claim 40, further comprising an instrument operably coupled to said computer wherein the graphical program is operable to invoke said method of said object for performing instrumentation functions on said instrument.

42. A system for creating a graphical automation client in a graphical program which is operable to invoke a property of an object exported by an automation server, comprising:

a computer including a display screen;

a user input device for receiving user input specifying an automation class;

an automation open node displayed on the screen in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

an automation property node displayed on the screen in response to user input, wherein said automation property node is operable to invoke a property of said object;

a wire displayed on the screen in response to user input for connecting said automation open node to said automation property node, wherein said automation open node is operable to provide information on said object to said automation property node;

wherein said automation property node receives said information on said object, wherein said automation property node is operable to invoke said property of said object during execution of said graphical program.

43. The system of claim 42, further comprising an instrument operably coupled to said computer wherein the graphical program is operable to invoke said property of said object for performing instrumentation functions on said instrument.

44. A system for creating a graphical automation client in a graphical program which is operable to invoke a method of an object exported by an automation server, comprising:

a computer including a display screen and a user input device;

means for receiving user input specifying an automation class;

means for displaying on the screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

means for displaying on the screen an automation invoke node in response to user input, wherein said automation invoke node is operable to invoke a method of said object;

means for connecting said automation open node to said automation invoke node in response to user input, wherein said automation open node is operable to provide information on said object to said automation invoke node;

wherein said automation invoke node receives said information on said object, wherein said automation invoke node is operable to invoke said method of said object during execution of said graphical program.

45. A system for creating a graphical automation client in a graphical program which is operable to invoke a property of an object exported by an automation server, comprising:

a computer including a display screen and a user input device;

means for receiving user input specifying an automation class;

means for displaying on the screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

means for displaying on the screen an automation property node in response to user input, wherein said automation property node is operable to invoke a property of said object;

means for connecting said automation open node to said automation property node in response to user input, wherein said automation open node is operable to provide information on said object to said automation property node;

wherein said automation property node receives said information on said object, wherein said automation property node is operable to invoke said property of said object during execution of said graphical program.

46. A system for creating a graphical automation client in a graphical program which is operable to invoke a function of an object exported by an automation server, comprising:

a computer including a display screen;

a user input device for receiving user input specifying an automation class;

an automation open node which is displayed on the screen in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

an automation function node displayed on the screen in response to user input, wherein said automation function node is operable to invoke a function of said object;

a wire which is displayed on the screen in response to user input for connecting said automation open node to said automation function node, wherein said automation open node is operable to provide information on said object to said automation function node;

wherein said automation function node receives said information on said object, wherein said automation function node is operable to invoke said function of said object during execution of said graphical program.

47. A computer-readable storage medium comprising program instructions, wherein said program instructions are operable to implement the steps of:

receiving user input specifying an automation class;

displaying on a screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

displaying on the screen an automation invoke node in response to user input, wherein said automation invoke node is operable to invoke a method of said object;

connecting said automation open node to said automation invoke node in response to user input, wherein said automation open node is operable to provide information on said object to said automation invoke node;

wherein said automation invoke node receives said information on said object, wherein said automation invoke node is operable to invoke said method of said object during execution of a graphical program comprising said program instructions.

48. The computer-readable storage medium of claim 47, further comprising program instructions operable to implement the steps of:

instantiating an object of said automation class and invoking said method of said object.

49. The computer-readable storage medium of claim 47, further comprising program instructions operable to implement the steps of:

displaying on the screen a list of type libraries associated with one or more automation servers present in a computer executing said program instructions;

selecting a type library from said list of type libraries in response to said receiving user input specifying said automation class, wherein said automation class is from said selected type library.

50. The computer-readable storage medium of claim 47, further comprising program instructions operable to implement the steps of:

displaying on the screen a list of possible automation classes from a type library associated with an automation server exporting said object;

selecting said automation class from said list of possible automation classes in response to said receiving user input specifying said automation class.

51. The computer-readable storage medium of claim 47, wherein said information on said object includes a reference to said object.

52. The computer-readable storage medium of claim 47, wherein said displaying on the screen said automation invoke node comprises selecting said method in response to user input and displaying on the screen said method in said automation invoke node.

53. The computer-readable storage medium of claim 47, wherein said method includes one or more parameters, wherein said displaying on the screen said automation invoke node comprises displaying a terminal for each of said one or more parameters.

54. The computer-readable storage medium of claim 53, wherein each said terminal for each of said one or more parameters has an associated data type.

55. The computer-readable storage medium of claim 47, wherein said automation class is an automation class of an automation technology selected from the list of automation technologies consisting of OLE, CORBA and OpenDoc.

56. The computer-readable storage medium of claim 47, further comprising program instructions operable to implement the steps of:

displaying a list of automation technologies;

selecting one of said automation technologies, in response to said receiving user input specifying said automation class, wherein said automation class conforms with said selected automation technology.

57. The computer-readable storage medium of claim 47, further comprising program instructions implementing the step of:

displaying a palette of automation nodes from which said automation open node and said automation invoke node may be selected.

58. The computer-readable storage medium of claim 47, wherein the program instructions are operable to invoke said method of said object for performing instrumentation functions on an instrument.

59. The computer-readable storage medium of claim 47, wherein said connecting said automation open node to said automation invoke node comprises. displaying on the screen a wire connecting said automation open node to said automation invoke node.

60. A computer-readable storage medium comprising program instructions, wherein said program instructions are operable to implement the steps of:

receiving user input specifying an automation class;

displaying on a screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

displaying on the screen an automation property node in response to user input, wherein said automation property node is operable to invoke a property of said object;

connecting said automation open node to said automation property node in response to user input, wherein said automation open node is operable to provide information on said object to said automation property node;

wherein said automation property node receives said information on said object, wherein said automation property node is operable to invoke said property of said object during execution of a graphical program comprising said program instructions.

61. The computer-readable storage medium of claim 60, further comprising program instructions operable to implement the steps of:

instantiating an object of said automation class and invoking said property of said object.

62. The computer-readable storage medium of claim 60, further comprising program instructions operable to implement the steps of:

displaying on the screen a list of type libraries associated with one or more automation servers present in a computer executing said program instructions;

selecting a type library from said list of type libraries in response to said receiving user input specifying said automation class, wherein said automation class is from said selected type library.

63. The computer-readable storage medium of claim 60, further comprising program instructions operable to implement the steps of:

displaying on the screen a list of possible automation classes from a type library associated with an automation server exporting said object;

selecting said automation class from said list of possible automation classes in response to said receiving user input specifying said automation class.

64. The computer-readable storage medium of claim 60, wherein said information on said object

includes a reference to said object.

65. The computer-readable storage medium of claim 60, wherein said displaying on the screen said automation property node comprises selecting said property in response to user input and displaying on the screen said property in said automation property node.

66. The computer-readable storage medium of claim 60, wherein said displaying on the screen said automation property node comprises displaying a terminal for said property.

67. The computer-readable storage medium of claim 66, wherein said terminal has an associated data type.

68. The computer-readable storage medium of claim 60, wherein said automation class is an automation class of an automation technology selected from the list of automation technologies consisting of OLE, CORBA and OpenDoc.

69. The computer-readable storage medium of claim 60, further comprising program instructions operable to implement the steps of:

displaying a list of automation technologies;

selecting one of said automation technologies, in response to said receiving user input specifying said automation class, wherein said automation class conforms with said selected automation technology.

70. The computer-readable storage medium of claim 60, further comprising program instructions implementing the step of:

displaying a palette of automation nodes from which said automation open node and said automation property node may be selected.

71. The computer-readable storage medium of claim 60, wherein the program instructions are operable to invoke said property of said object for performing instrumentation functions on an instrument.

72. The computer-readable storage medium of claim 60, wherein said connecting said automation open node to said automation property node comprises displaying on the screen a wire connecting said automation open node to said automation property node.

73. A computer-readable storage medium comprising program instructions, wherein said program instructions are operable to implement the steps of:

receiving user input specifying an automation class;

displaying on a screen an automation open node in response to user input, wherein said automation open node is operable to create an instance of said automation class, wherein said instance of said automation class is an object;

displaying on the screen an automation function node in response to user input, wherein said

automation function node is operable to invoke a function of said object;

connecting said automation open node to said automation function node in response to user input, wherein said automation open node is operable to provide information on said object to said automation function node;

wherein said automation function node receives said information on said object, wherein said automation function node is operable to invoke said function of said object during execution of said graphical program comprising said program instructions.

74. The computer-readable storage medium of claim 73, wherein said automation function node comprises an automation invoke node and said function comprises a method of said object.

75. The computer-readable storage medium of claim 73, wherein said automation function node comprises an automation property node and said function comprises a property of said object.

76. A computer-implemented method for creating a graphical program, wherein the graphical program is operable to invoke a method of an object, wherein the method for creating the graphical program operates in a computer including a display screen and a user input device, the method for creating the graphical program comprising:

displaying on the screen an invoke node in the graphical program in response to user input, wherein said invoke node is operable to invoke a method of an object;

configuring said invoke node to receive information on said object in response to user input;

wherein, during execution of the graphical program, said invoke node is operable to invoke said method of said object.

77. The computer-implemented method of claim 76, wherein said invoke node includes an object reference input for receiving a reference to the object;

wherein said configuring comprises connecting said object reference input of said invoke node to receive the reference to said object;

wherein said invoke node receives said information on said object on said object reference input during execution of the graphical program.

78. The computer-implemented method of claim 77, wherein said configuring comprises:

displaying on the screen an object reference node which includes an object reference output that provides the reference to said object; and

connecting the object reference output of said object reference node to said object reference input of said invoke node.

79. The computer-implemented method of claim 76, wherein said configuring comprises configuring said invoke node with a reference to said object in response to user input.

80. The computer-implemented method of claim 76, wherein the object is comprised in a server, wherein said configuring comprises:

displaying on the screen a list of libraries associated with one or more servers;

selecting a library from said list of libraries in response to user input

displaying on the screen a list of possible classes from the selected library;

selecting a class from said list of possible classes in response to user input;

wherein said object is instantiated from said class.

81. The computer-implemented method of claim 76, further comprising:

constructing execution instructions in response to said graphical program, wherein said execution instructions are executable to invoke said method of said object; and

executing said execution instructions, wherein said invoke node invokes said method of said object during said executing.

82. A computer-implemented method for creating a graphical program, wherein the graphical program is operable to invoke a property of an object, wherein the method for creating the graphical program operates in a computer including a display screen and a user input device, the method for creating the graphical program comprising:

displaying on the screen a property node in the graphical program in response to user input, wherein said property node is operable to invoke a property of an object;

configuring said property node to receive information on said object in response to user input;

wherein, during execution of the graphical program, said property node is operable to invoke said property of said object.

83. The computer-implemented method of claim 82, wherein said property node includes an object reference input for receiving a reference to the object;

wherein said configuring comprises connecting said object reference input of said property node to receive the reference to said object;

wherein said property node receives said information on said object on said object reference input during execution of the graphical program.

84. The computer-implemented method of claim 83, wherein said configuring comprises:

displaying on the screen an object reference node which includes an object reference output that provides the reference to said object; and

connecting the object reference output of said object reference node to said object reference input of said property node.

85. The computer-implemented method of claim 82, wherein said configuring comprises configuring said property node with a reference to said object in response to user input.

86. The computer-implemented method of claim 82, wherein the object is comprised in a server, wherein said configuring comprises:

displaying on the screen a list of libraries associated with one or more servers;

selecting a library from said list of libraries in response to user input

displaying on the screen a list of possible classes from the selected library;

selecting a class from said list of possible classes in response to user input;

wherein said object is instantiated from said class.

87. The computer-implemented method of claim 82, further comprising:

constructing execution instructions in response to said graphical program, wherein said execution instructions are executable to invoke said property of said object; and

executing said execution instructions, wherein said property node invokes said property of said object during said executing.

88. The computer-implemented method of claim 82, wherein said property node is operable to get and/or set one or more properties of said object.

89. The computer-implemented method of claim 76, wherein the graphical program is operable to invoke said method of said object for performing instrumentation functions on an instrument.

90. The computer-implemented method of claim 82, wherein the graphical program is operable to invoke said property of said object for performing instrumentation functions on an instrument.

First Hit Fwd Refs

Generate Collection Print

L1: Entry 163 of 314 File: USPT Aug 20, 2002

US-PAT-NO: 6437805
DOCUMENT-IDENTIFIER: US 6437805 B1

TITLE: System and method for accessing object capabilities in a graphical program

DATE-ISSUED: August 20, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Sojoodi; Omid	Austin	TX		
Dye; Robert	Austin	TX		
Parthasarathy; Murali	Austin	TX		
Kudukoli; Ram	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
National Instruments Corporation	Austin	TX			02

APPL-NO: 09/ 136123 [PALM]
DATE FILED: August 18, 1998

PARENT-CASE:
PRIORITY INFORMATION This application claims the benefit of priority of: U.S. Provisional Application No. 60/056,528 titled "System and Method for Providing Automation Server Capabilities in Graphical Programs," by Ram Kudukoli, Robert Dye and Murali Parthasarathy, filed on Aug. 21, 1997; CONTINUATION DATA This is a continuation-in-part of co-pending patent application Ser. No. 08/916,005 titled "System and Method for Providing Client/Server Access to Graphical Programs" filed on Aug. 21, 1997, whose inventors were Robert Dye and Omid Sojoodi which issued Aug. 15, 2000 as U.S. Pat. No. 6,102,965 which is a continuation in part of co-pending patent application Ser. No. 08/810,079 titled "System and Method for Developing Automation Clients using a Graphical Data Flow Program" filed on Mar. 4, 1997, whose inventors were Murali Parthasarathy and Omid Sojoodi, which issued May 16, 2000 as U.S. Pat. No. 6,064,812 which is a continuation-in-part of co-pending application Ser. No. 08/717,771 which issued Dec. 8, 1998 as U.S. Pat. No. 5,847,953 titled "System and Method for Performing Class Checking of Objects in a Graphical Data Flow Program" and filed Sep. 23, 1996, whose inventors were Omid Sojoodi and Stephen W. Rogers, which issued Dec. 8,1998 as U.S. Pat. No. 5,847,953. This is also a continuation in part of co-pending patent application Ser. No. 08/810,079 titled "System and Method for Developing Automation Clients using a Graphical Data Flow Program" filed on Mar. 4, 1997, whose inventors were Murali Parthasarathy and Omid Sojoodi, which issued May. 16, 2000 as U.S. Pat. No. 6,064,812 which is a continuation-in-part of co-pending application Ser. No. 08/717,771 titled "System and Method for Performing Class Checking of Objects in a Graphical Data Flow Program" and filed Sep. 23, 1996, whose inventors were Omid Sojoodi and Stephen W. Rogers which issued Dec. 8, 1998 as U.S. Pat. No. 5,847,953. This is also a continuation-in-part of co-pending application Ser. No. 08/717,771 titled "System and Method for Performing Class Checking of Objects in a Graphical

Data Flow Program" and filed Sep. 23, 1996, whose inventors were Omid Sojoodi and Stephen W. Rogers which issued Dec. 8, 1998 as U.S. Pat. No. 5,847,953. CROSS-REFERENCE TO RELATED APPLICATIONS The following applications are related to the present application: U.S. patent application Ser. No. 08/916,005 titled "System and Method for Providing Client/Server Access to Graphical Programs" filed on Aug. 21, 1997, whose inventors were Robert Dye and Omid Sojoodi which issued Aug. 15, 2000 as U.S. Pat. No. 6,102,965; U.S. patent application Ser. No. 08/810,079 titled "System and Method for Developing Automation Clients using a Graphical Data Flow Program" filed on Mar. 4, 1997, whose inventors were Murali Parthasarathy and Omid Sojoodi, which issued May. 16, 2000 as U.S. Pat. No. 6,064,812, U.S. patent application Ser. No. 08/811,187 titled "System and Method for Performing Class Propagation and Type Checking in a Graphical Automation Client" and filed on Mar. 4, 1997, whose inventors were Murali Parthasarathy and Omid Sojoodi, which issued May. 16, 2000 as U.S. Pat. No. 6,064,812, U.S. patent application Ser. No. 08/717,771 titled "System and Method for Performing Class Checking of Objects in a Graphical Data Flow Program" which issued Dec. 8, 1998 as U.S. Pat. No. 5,847,953 Stephen W. Rogers. U.S. patent application Ser. No. 08/717,772 titled "System and Method for Performing Interface Independent Virtual Instrumentation Functions Using Attribute Nodes in a Graphical Data Flow Program" which issued May. 18, 2000 as U.S. Pat. No. 5,905,648 and filed Sep. 23, 1996, whose inventors were Omid Sojoodi and Stephen W. Rogers.

INT-CL: [07] G09 G 5/00

US-CL-ISSUED: 345/763; 717/105
US-CL-CURRENT: 345/763; 717/105

FIELD-OF-SEARCH: 345/333, 345/335, 345/349, 345/339, 345/334, 345/967, 345/440, 345/357, 345/835, 345/765, 345/744, 345/854, 345/619, 345/763, 345/764, 717/1, 717/104-109, 717/132, 709/225, 709/220

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL | Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4901221	February 1990	Kodosky et al.	
☐	5097411	March 1992	Doyle et al.	345/522
☐	5251322	October 1993	Doyle et al.	345/501
☐	5261043	November 1993	Wolber et al.	
☐	5291587	March 1994	Kodosky et al.	703/2
☐	5337262	August 1994	Luthi et al.	
☐	5359546	October 1994	Hayes et al.	
☐	5361336	November 1994	Atchison	
☐	5390325	February 1995	Miller	
☐	5394522	February 1995	Sanchez-Frank et al.	345/349
☐	5428776	June 1995	Rothfield	707/4
☐	5481741	January 1996	McKaskle et al.	

☐	5513311	April 1996	McKiel, Jr.	395/161
☐	5576946	November 1996	Bender et al.	
☐	5640572	June 1997	Mondrik et al.	
☐	5673403	September 1997	Brown et al.	395/335
☐	5727175	March 1998	Malone et al.	345/356
☐	5734863	March 1998	Kodosky et al.	703/27
☐	5751914	May 1998	Coley et al.	395/500.43
☐	5758071	May 1998	Burgess et al.	395/200.5
☐	5758084	May 1998	Silverstein et al.	395/200.58
☐	5768578	June 1998	Kirk et al.	707/100
☐	5784275	July 1998	Sojoodi et al.	
☐	5784583	July 1998	Redpath	345/353
☐	5801942	September 1998	Nixon et al.	700/83
☐	5802514	September 1998	Huber	395/701
☐	5802526	September 1998	Fawcett et al.	707/104
☐	5812133	September 1998	Schultz et al.	345/347
☐	5821934	October 1998	Kodosky et al.	
☐	5847953	December 1998	Sojoodi et al.	364/188
☐	5848273	December 1998	Fontana et al.	395/701
☐	5850548	December 1998	Williams	717/1
☐	5862339	January 1999	Bonnamre et al.	395/200.57
☐	5862379	January 1999	Rubin et al.	395/702
☐	5867665	February 1999	Butman et al.	395/200.68
☐	5895474	April 1999	Maarek et al.	707/514
☐	5905649	May 1999	Sojoodi et al.	700/83
☐	5966532	October 1999	McDonald et al.	717/1
☐	6064812	May 2000	Parthasarathy et al.	717/1
☐	6102965	August 2000	Dye et al.	717/1

OTHER PUBLICATIONS

Rijnders et al., "Versatile Visual Programming Environment for Scientific Applications," ACM, 1991, pp. 21-26.*
Paul S. Barth, "An Object-Oriented Approach to Graphical Interfaces," ACM Transactions on Graphics, vol. 5, No. 2, 1986, pp 142-172.*
LabVIEW Function Reference Manual, Aug. 1993 Edition, pp 10-11 and 10-12.
LabVIEW Code Interface Reference Manual, Dec. 1993.
Pleas, K., BYTE Magazine Special Report, Apr. 1996, "OLE's Missing Links: An emerging standard for communication between applications, OLE 2 will work better when software vendors agree on its meaning," 6 pages.
Linthicum, D., BYTE Magazine State of the Art, Jan. 1996, "Integration, Not Perspiration: CORBA, OLE, and OpenDoc: Three technologies for desktop components face off," 11 pages.

LabVIEW2 User Manual, Jan. 1990, National Instruments Corp., pp 3-18 to 3-19.
Smedley, Trevor J., "Visual Programming Applied to Industrial Software Development", 1995, pp. 2-28.
Hewlett Packard SICL Standard Instrument Control Library for C Progamming, Lee Atchison, VXI Systems Division, Hewlett Packard Company, Jan. 21, 1994, Revision 3.9, pp. i-iii, 1-136, Index.

ART-UNIT: 2173

PRIMARY-EXAMINER: Bayerl; Raymond J.

ASSISTANT-EXAMINER: Hailu; Tadesse

ATTY-AGENT-FIRM: Conley, Rose & Tayon PC Hood; Jeffrey C.

ABSTRACT:

A system and method for creating a graphical program, wherein the graphical program is operable to access capabilities of an object. During creation of the graphical program, the user operates to place an object node in the graphical program, wherein the object node is operable to access capabilities of the object. This preferably includes the user arranging on the screen the graphical program, including the object node and various other nodes, and connecting the various nodes to create the graphical program. The user then configures the object node to receive information on the object, preferably by the user configuring the object node with a reference to the object, e.g., a pointer, address, or other information which specifies the identity and/or location of the object. The user also selects one or more methods to be invoked on the object and/or one or more properties to get/set on the object. Once the graphical program has been created, then during execution of the graphical program, the object node accesses the capabilities of the object.

84 Claims, 63 Drawing figures

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: (graphical$2 near4 (data adj flow))

Display: 40 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search | Clear | Interrupt

Search History

DATE: Thursday, March 04, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ			
L2	L1 and (flow?.ti. or flow?.ab.)	2	L2
L1	(graphical$2 near4 (data adj flow))	314	L1

Scanned all

END OF SEARCH HISTORY

L1: Entry 269 of 314 File: USPT Jan 2, 1996

DOCUMENT-IDENTIFIER: US 5481741 A
TITLE: Method and apparatus for providing attribute nodes in a graphical data flow environment

Brief Summary Text (3):
The present invention relates to graphical systems for creating and executing data flow programs, and more specifically to a method and apparatus for providing attribute nodes in a graphical data flow environment.

Brief Summary Text (17):
Therefore, Kodosky et al teaches a graphical programming environment wherein a user manipulates icons in a block diagram using a block diagram editor to create a data flow "program" or virtual instrument (VI). In creating a virtual instrument, a user first creates a front panel including various controls or indicators that represent the respective input and output that will be used by the VI. When the controls and indicators are created in the front panel, corresponding icons or terminals are automatically created in the block diagram by the block diagram editor. The user then chooses various functions that accomplish his desired result, connecting the corresponding function icons between the terminals of the respective controls and indicators. In other words, the user creates a data flow program, referred to as a block diagram, representing the graphical data flow which accomplishes his desired function. This is done by wiring up the various function icons between the control icons and indicator icons. The manipulation and organization of icons in turn produces machine language that accomplishes the desired method or process as shown in the block diagram. A user then optionally chooses a connector pane representing the input and output terminals corresponding to the respective controls and indicators already created.

Brief Summary Text (19):
A user inputs data to a virtual instrument using front panel controls. This input data propagates through the data flow block diagram or graphical program and appears as changes on the output indicators. The data that flows from the controls to the indicators in this manner is referred to as control data. In an instrumentation application, the front panel can be analogized to the front panel of an instrument. The user adjusts the controls on the front panel to affect the input and views the output on the respective indicators.

Detailed Description Text (189):
In instrumentation applications according to the preferred embodiment, graphical data flow programs are referred to as virtual instruments (VIs). As previously discussed VIs have three main parts: the front panel, the block diagram, and the icon/connector.

Detailed Description Text (291):
The Unopened Type Defs option . . . allows the user to open a file containing the original of a type definition. For more information please see related U.S. patent application Ser. No. 0/125,459, titled "Method and Apparatus for Providing Improved Type Compatability Checking Data Structure Variable Referencing in a Graphical Data Flow Program," filed Sep. 22, 1993, which is hereby incorporated by reference.

Detailed Description Text (412):
To summarize, in creating a virtual instrument, a user first creates a front panel including various controls or indicators that represent the respective input and output that will be used by the VI. When the controls and indicators are created in the front panel, corresponding icons are automatically created in the block diagram by the block diagram editor. The user then chooses various functions, etc. that accomplish his desired result, connecting these function icons between the terminals of the respective control icons and indicator icons. In other words, the user creates a block diagram representing the graphical data flow program which accomplishes his desired function. This is done by wiring up various functions between the control icons and indicator icons. The manipulation and organization of icons in turn produces machine language code that accomplishes the desired method or process as shown in the block diagram. A user then optionally chooses a connector pane representing the input and output terminals corresponding to the respective controls and indicators already created.

Detailed Description Text (414):
A user adjusts the input to the virtual instrument by changing data input using the controls. This input data from the user propagates through the data flow block diagram or graphical program and appears as changes on the indicators. The data that flows from the controls to the indicators in this manner is referred to as control data.

CLAIMS:

1. A computer implemented method for programmatically affecting an attribute of a control in a data flow program in a computer system including a video screen, means for creating a graphical data flow diagram, and means for creating a panel associated with said data flow diagram for displaying data input to and output from said data flow diagram, the method comprising the computer implemented steps of:

displaying on the screen a first panel;

displaying on the screen a first control which displays data, wherein said first control is comprised in said first panel;

displaying on the screen a first function icon that references a function icon control means for controlling a first function;

displaying on the screen an attribute node icon associated with said first control, wherein the attribute node icon references an attribute control means for programmatically affecting an attribute of said first control;

assembling on the screen a data flow diagram including the first function icon and the attribute node icon, wherein the first function icon is connected to the attribute node icon and wherein the function icon control means provides data to the attribute control means during execution of the data flow diagram, wherein said first panel is associated with said data flow diagram and wherein said first control in said first panel displays input or output data from said data flow diagram;

executing the data flow diagram;

the function icon control means writing a value to the attribute control means to affect said attribute of said first control during said step of executing; and

changing said attribute of said first control after said step of the function icon control means writing said value to the attribute control means to affect said attribute of said first control.

12. A computer implemented method for programmatically accessing an attribute of a control in a data flow program in a computer system including a video screen, means for creating a graphical data flow diagram, and means for creating a panel associated with said data flow diagram for displaying data input to and output from said data flow diagram, the method comprising the computer implemented steps of:

displaying on the screen a first panel;

displaying on the screen a first control which displays data, wherein said first control is comprised in said first panel;

displaying on the screen a first function icon that references a function icon control means for controlling a first function;

displaying on the screen an attribute node icon associated with said first control, wherein the attribute node icon references an attribute control means for programmatically accessing an attribute of said first control;

assembling on the screen a data flow diagram including the first function icon and the attribute node icon, wherein the attribute node icon is connected to the first function icon and wherein the attribute control means provides data to the function icon control means during execution of the data flow diagram, wherein said first panel is associated with said data flow diagram and wherein said first control in said first panel displays input or output data from said data flow diagram;

executing the data flow diagram;

receiving input from a user during said step of executing to change said attribute of said first control;

the attribute control means reading said attribute of said first control and generating a value indicative thereof during said step of executing;

the attribute control means providing said value to said function icon control means during said step of executing; and

the function icon control means computing a value using said value received from the attribute control means during said step of executing.

24. A computer implemented method for programmatically accessing an attribute of a control in a data flow program in a computer system including a video screen, means for creating a graphical data flow diagram, and means for creating a panel associated with said data flow diagram for displaying data input to and output from said data flow diagram, the method comprising the computer implemented steps of:

displaying on the screen a first panel;

displaying on the screen a first control which displays data, wherein said first control is comprised in said first panel;

displaying on the screen a first function icon that references a first function icon control means for controlling a first function;

displaying on the screen a second function icon that references a second function icon control means for controlling a second function;

displaying on the screen an attribute node icon associated with said first control, wherein the attribute node icon references an attribute control means for programmatically accessing an attribute of said first control;

assembling on the screen a data flow diagram including the first function icon, the second function icon, and the attribute node icon, wherein the first function icon is connected to the attribute node icon and wherein the attribute node icon is connected to the second function icon, wherein said first panel is associated with said data flow diagram and wherein said first control in said first panel displays input or output data from said data flow diagram;

executing the data flow diagram;

the first function icon control means writing a value to the attribute control means to affect said attribute of said first control during said step of executing; and

the second function icon control means reading a value from the attribute control means to access said attribute of said first control during said step of executing.

33. A computer implemented method for programming a computer system including a video screen, means for creating a graphical data flow diagram, and means for creating a panel associated with said data flow diagram for displaying data input to and output from said data flow diagram, the method comprising the computer implemented steps of:

displaying on the screen a first panel;

displaying on the screen a first control in said first panel, wherein said first control displays data;

displaying on the screen a second control in said first panel, wherein said second control displays data;

displaying on the screen a first terminal icon that references said first control;

displaying on the screen a second terminal icon that references said second control;

displaying on the screen a first function icon that references a function icon control means for controlling a first function;

displaying on the screen an attribute node icon associated with said first control that references an attribute control means for programmatically affecting an attribute of said first control;

displaying on the screen a connector pane icon having a plurality of terminals for linking said first and second terminal icons, wherein said connector pane icon has associated connector control means for linking controls;

assigning said first control to a first terminal of said connector pane icon;

assigning said second control to a second terminal of said connector pane icon;

assembling on the screen a data flow diagram including the first terminal icon and the second terminal icon and the first function icon and the attribute node icon, wherein said first and second controls display data in said data flow diagram;

executing the data flow diagram;

propagating control data from said first control to said second control in a first data structure during said step of executing; and

propagating attribute data between said attribute control means and said second control using a second data structure.